Unit - 1 (Introduction, Asymptotic Notations, Recursive functions)

Ticking Session-3

Name: Nikhil Girish
SRN: PES2UG21CS334
Section: 4F

1. A matrix is given with nxn dimensions, each row is treated as a binary number and the value of the matrix is given as the sum of these binary numbers. Adarsh wants to maximise this value of the matrix, he is allowed to choose any row or column and toggle each value in that row or column(i.e changing all 0's to 1's, and all 1's to 0's in that row or column). Find the max value of the given matrix after making necessary toggles.

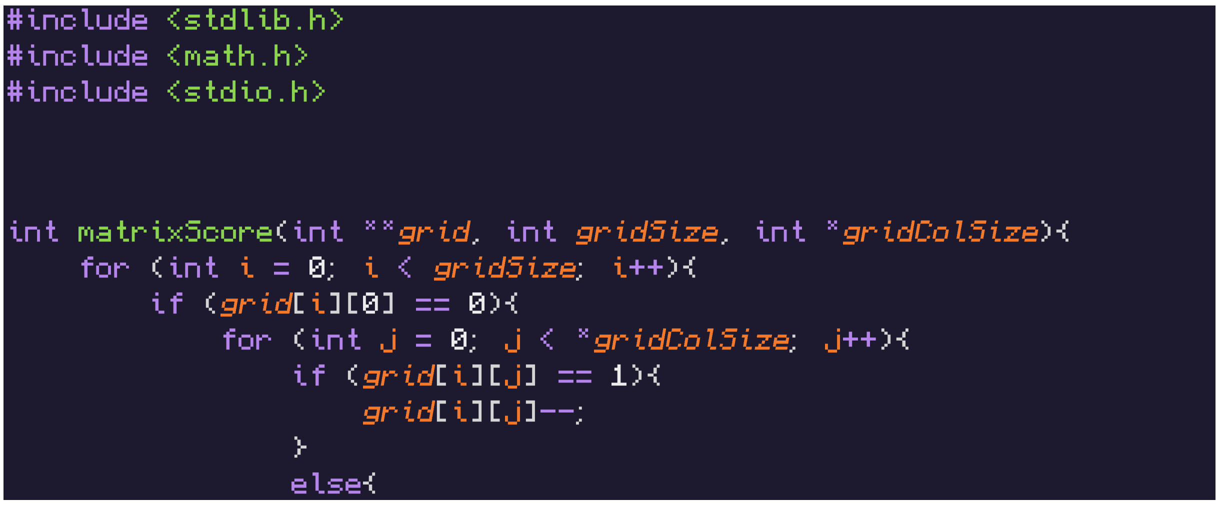
Sample Input 1:

```
3 4                // Order of matrix
0 0 1 1
1 0 1 0           // Elements of matrix
1 1 0 0
```

Sample Output 1:

Output -
39

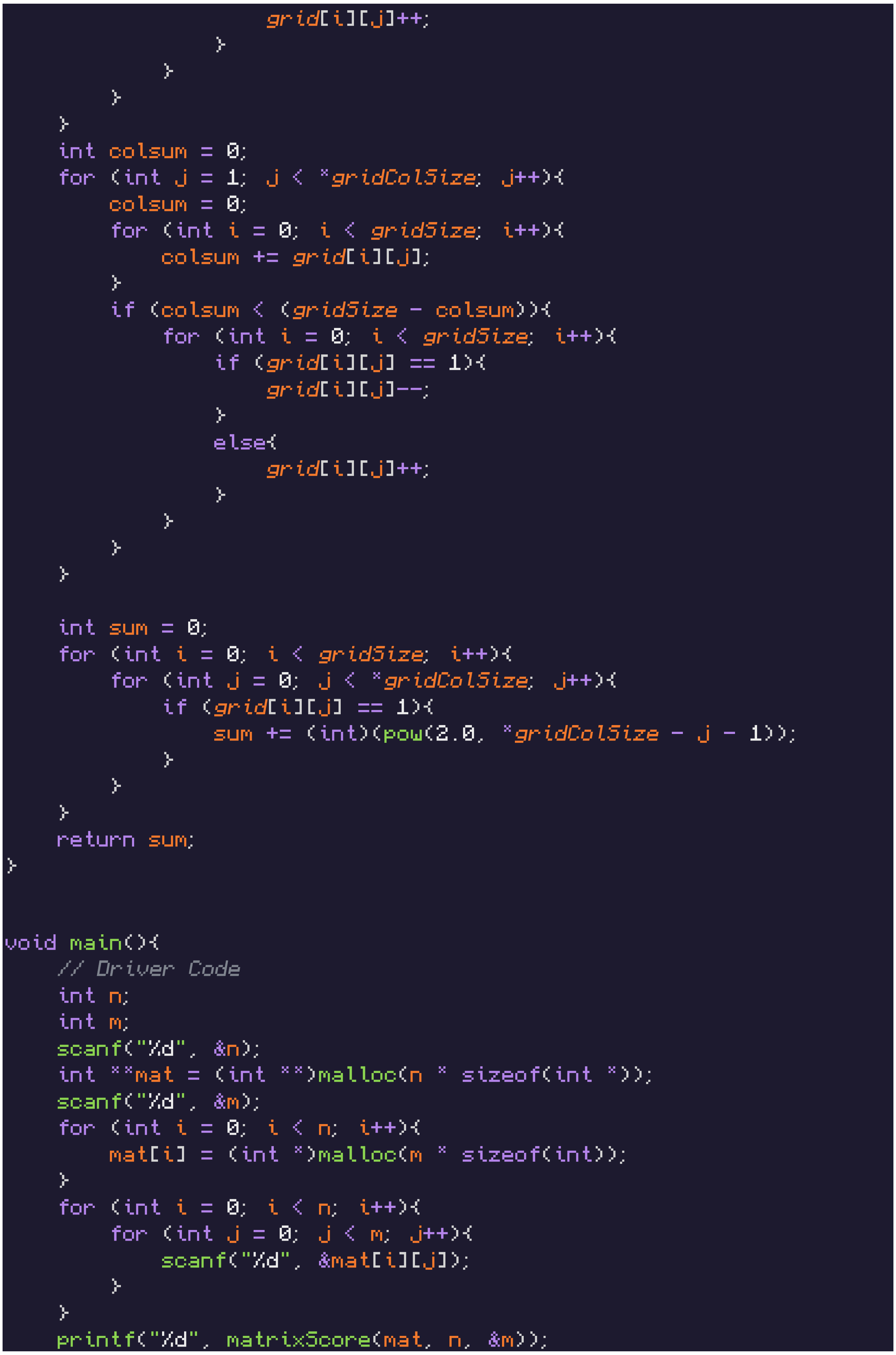
Boilerplate/Skeleton Code:



```
#include <stdlib.h>
#include <math.h>
#include <stdio.h>



int matrixScore(int **grid, int gridSize, int *gridColSize){
    for (int i = 0; i < gridSize; i++){
        if (grid[i][0] == 0){
            for (int j = 0; j < *gridColSize; j++){
                if (grid[i][j] == 1){
                    grid[i][j]--;
                }
                else{
```



```c
                    grid[i][j]++;
                }
            }
        }
    }
    int colsum = 0;
    for (int j = 1; j < *gridColSize; j++){
        colsum = 0;
        for (int i = 0; i < gridSize; i++){
            colsum += grid[i][j];
        }
        if (colsum < (gridSize - colsum)){
            for (int i = 0; i < gridSize; i++){
                if (grid[i][j] == 1){
                    grid[i][j]--;
                }
                else{
                    grid[i][j]++;
                }
            }
        }
    }

    int sum = 0;
    for (int i = 0; i < gridSize; i++){
        for (int j = 0; j < *gridColSize; j++){
            if (grid[i][j] == 1){
                sum += (int)(pow(2.0, *gridColSize - j - 1));
            }
        }
    }
    return sum;
}


void main(){
    // Driver Code
    int n;
    int m;
    scanf("%d", &n);
    int **mat = (int **)malloc(n * sizeof(int *));
    scanf("%d", &m);
    for (int i = 0; i < n; i++){
        mat[i] = (int *)malloc(m * sizeof(int));
    }
    for (int i = 0; i < n; i++){
        for (int j = 0; j < m; j++){
            scanf("%d", &mat[i][j]);
        }
    }
    printf("%d", matrixScore(mat, n, &m));
```

```
}
```

Test cases:

1.

1 1 // Order of matrix

0

Output-
1

2.

2 1 // Order of matrix

0
1

Output-
2

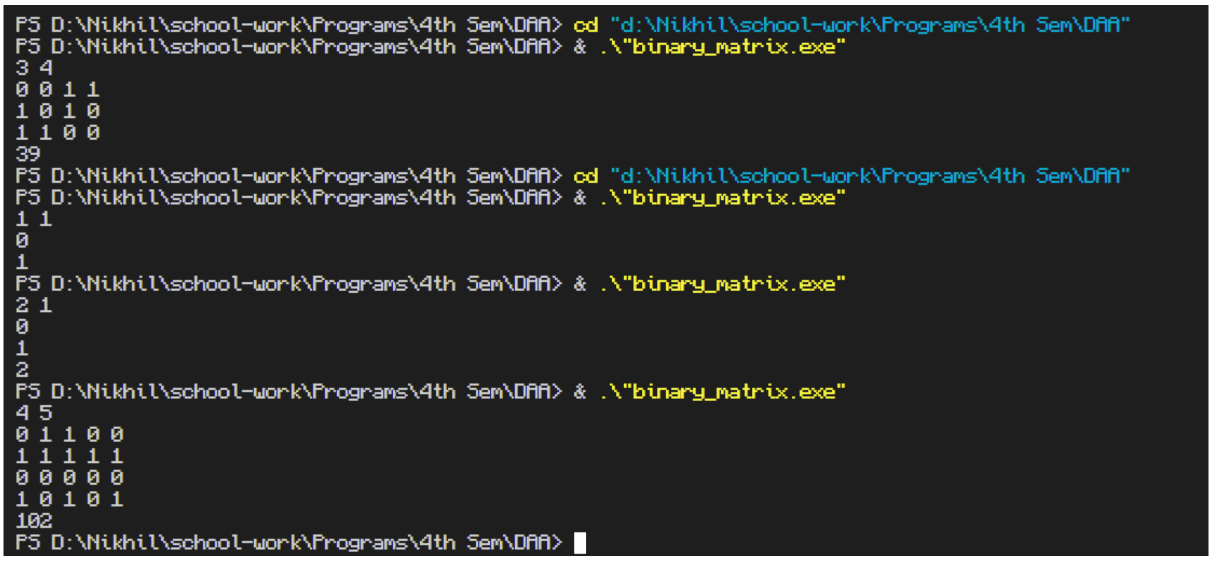
3.

4 5 // Order of matrix

0 1 1 0 0
1 1 1 1 1
0 0 0 0 0
1 0 1 0 1

Output-
102

Output Screenshots:



```
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> cd "d:\Nikhil\school-work\Programs\4th Sem\DAA"
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> & .\"binary_matrix.exe"
3 4
0 0 1 1
1 0 1 0
1 1 0 0
39
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> cd "d:\Nikhil\school-work\Programs\4th Sem\DAA"
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> & .\"binary_matrix.exe"
1 1
0
1
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> & .\"binary_matrix.exe"
2 1
0
1
2
PS D:\Nikhil\school-work\Programs\4th Sem\DAA> & .\"binary_matrix.exe"
4 5
0 1 1 0 0
1 1 1 1 1
0 0 0 0 0
1 0 1 0 1
102
PS D:\Nikhil\school-work\Programs\4th Sem\DAA>
```